UNCLASSIFIED

IDA MEMORANDUM REPORT M-454

12

SOURCES OF COMPILER CAPABILITY INFORMATION IN VALIDATION SUMMARY REPORTS

R. Danford Lehman
Audrey A. Hook
James Wolfe

May 1988

Prepared for
Ada Joint Program Office

INSTITUTE FOR DEFENSE ANALYSES
1801 N. Beauregard Street, Alexandria, Virginia 22311

UNCLASSIFIED

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETEING FORM

1. REPORT NUMBER | 2. GOVT ACCESSION NO. | 3. RECIPIENT'S CATALOG NUMBER

4. TITLE *(and Subtitle)*
Sources of Compiler Capability information in Validation Summary Reports

5. TYPE OF REPORT & PERIOD COVERED

6. PERFORMING ORG. REPORT NUMBER
IDA Memorandum Report M-454

7. AUTHOR(s)
R. Danford Lehman
Audrey A. Hook
James Wolfe

8. CONTRACT OR GRANT NUMBER(s)

9. PERFORMING ORGANIZATION AND ADDRESS
Institute for Defense Analysis
1081 N. Beauregard St.
Alexandris, VA 22311

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
T D5 304

11. CONTROLLING OFFICE NAME AND ADDRESS
Ada Joint Program Office
United States Department of Defense
Washington, DC 20301-3081

12. REPORT DATE
May 1988

13. NUMBER OF PAGES
24 p.

14. MONITORING AGENCY NAME & ADDRESS*(If different from Controlling Office)*
AJPO

15. SECURITY CLASS *(of this report)*
UNCLASSIFIED

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
N/A

16. DISTRIBUTION STATEMENT *(of this Report)*
Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20. If different from Report)*
UNCLASSIFIED

18. SUPPLEMENTARY NOTES

19. KEYWORDS *(Continue on reverse side if necessary and identify by block number)*
Ada Programming Language; Compilers; Validation Summary Report (VSR); Validation; Ada Compiler Validation Capability (ACVC); Ada Validation Facility (AVF); ACVC Test Suite; Implementation; ISO WG9.

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*
The purpose of IDA Memorandum Report M454 is to document the results of the Institute for Defense Analyses"s review of the Validation Summary Reports (VSRs). It is intended for use by the Ada Joint Program office staff concerned with porviding guidance to the DoD Program Managers on the use of VSRs as an information resource. The results of the formal validation of Ada compilers are documented in VSRs which are available through the National Technical Information Service (NTIS) to the public. The format and content of the VSR has been revised periodically to improve the presentation of data and to add substantive information concerning the testing procedures, configurations tested and testing results. The VSR is reviewed as a source of information on the capability of compilers that could be useful to a prospective buyer and user of an Ada

DD FORM 1473 1 JAN 73 S/N 0102-LF-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

DEFINITIONS

IDA publishes the following documents to report the results of its work.

Reports

Reports are the most authoritative and most carefully considered products IDA publishes. They normally embody results of major projects which (a) have a direct bearing on decisions affecting major programs, or (b) address issues of significant concern to the Executive Branch, the Congress and/or the public, or (c) address issues that have significant economic implications. IDA Reports are reviewed by outside panels of experts to ensure their high quality and relevance to the problems studied, and they are released by the President of IDA.

Papers

Papers normally address relatively restricted technical or policy issues. They communicate the results of special analyses, interim reports or phases of a task, ad hoc or quick reaction work. Papers are reviewed to ensure that they meet standards similar to those expected of refereed papers in professional journals.

Memorandum Reports

IDA Memorandum Reports are used for the convenience of the sponsors or the analysts to record substantive work done in quick reaction studies and major interactive technical support activities; to make available preliminary and tentative results of analyses or of working group and panel activities; to forward information that is essentially unanalyzed and unevaluated; or to make a record of conferences, meetings, or briefings, or of data developed in the course of an investigation. Review of Memorandum Reports is suited to their content and intended use.

The results of IDA work are also conveyed by briefings and informal memoranda to sponsors and others designated by the sponsors, when appropriate.

The work reported in this document was conducted under contract MDA 903 84 C 0031 for the Department of Defense. The publication of this IDA document does not indicate endorsement by the Department of Defense, nor should the contents be construed as reflecting the official position of that agency.

This Memorandum Report is published in order to make available the material it contains for the use and convenience of interested parties. The material has not necessarily been completely evaluated and analyzed, nor subjected to IDA review.

IDA MEMORANDUM REPORT M-454

SOURCES OF COMPILER CAPABILITY INFORMATION IN VALIDATION SUMMARY REPORTS

R. Danford Lehman
Audrey A. Hook
James Wolfe

May 1988

INSTITUTE FOR DEFENSE ANALYSES

Contract MDA 903 84 C 0031
Task T-D5-304

CONTENTS

LIST OF TABLES

1. INTRODUCTION

1.1 PURPOSE

The purpose of this IDA Memorandum Report is to document the result of IDA's review of Validation Summary Reports (VSRs) and is intended for use by the Ada Joint Program Office (AJPO) staff concerned with providing guidance to Department of Defense (DoD) Program Managers on the use of VSRs as an information resource. This report partially fulfills the requirements of Task T-D5-304, Ada Validation.

The results of formal validation of Ada compilers are documented in a VSR which is available through the National Technical Information Service (NTIS) to the public. The format and content of the VSR has been revised periodically to improve the presentation of data and to add substantive information concerning the testing procedures, configurations tested, and testing results. The AJPO requested that Institute for Defense Analyses (IDA) review the VSR as a source of information on the capability of compilers that could be useful to a prospective buyer and user of an Ada compiler.

1.2 SCOPE

The information provided in this report is based on analysis of information describing compiler capabilities generated from the validation process. Recommendations concern the status of the VSR as a source of compiler capability information.

The primary resources used in the preparation of this report include the Ada Compiler Validation Capability (ACVC) test suite, the *Reference Manual for the Ada Programming Language* (hereafter refered to as the LRM), and the *Validation Summary Report Template*. Documents containing compiler capability information for specific compilers include the VSR and the LRM's Appendix F which is required to accompany the documents describing a compiler. The VSRs from ACVC 1.8 and 1.9 were examined to determine current practices in the form and content of Appendix F.

1.3 REFERENCES

The following documents were referenced in this IDA Memorandum Report:

- Goodenough, John. *The Ada Compiler Validation Capability Implementor's Guide, Version 1.* Waltham, MA: SofTech, Inc., December 1986.
- International Standards Organization, WG 9. *ARG Commentary AI-00361.* Also known as AI-361.
- U.S. Department of Defense. *ANSI/MIL-STD-1815A, Reference Manual for the Ada Programming Language*. Washington, D.C.: DoD, 1983.

1.4 GLOSSARY

The following acronyms are used in this document.

ACVC Ada Compiler Validation Capability

AIG ACVC Implementor's Guide

AJPO Ada Joint Program Office

AVF Ada Validation Facility

DoD Department of Defense

IDA Institute for Defense Analyses

ISO International Standards Organization

LRM Reference Manual for the Ada Programming Language

NA Not Applicable

VSR Validation Summary Report

WG Working Group

2. BACKGROUND DISCUSSION

The ACVC 1.9 is a set of 3,122 tests designed to ensure that a submitted compiler conforms to the LRM. The suite is divided into the following six classes of tests known as the A, B, C, D, E, and L tests as described in the *Validation Summary Report Template*:

- Class A tests check that legal Ada programs can be successfully compiled and executed. However, no checks are performed during execution to see if the test objective has been met. For example, a Class A test checks that reserved words of another language (other than those already reserved in the Ada language) are not treated as reserved words by an Ada compiler.
- Class B tests check that a compiler detects illegal language usage. Class B tests are not executable. Each test in this class is compiled and the resulting compilation listing is examined to verify that every syntax or semantic error in the test is detected.
- Class C tests check that legal Ada programs can be correctly compiled and executed. Each Class C test is self-checking and produces a PASSED, FAILED, or NOT APPLICABLE message, indicating the result when it is executed.
- Class D tests check the compilation and execution capacities of a compiler. Since there are no capacity requirements placed on a compiler by the LRM for some parameters – for example, the number of identifiers permitted in a compilation or the number of units in a library – a compiler may refuse to compile a Class D test and still be a conforming compiler. Therefore, if a Class D test fails to compile because the capacity of the compiler is exceeded, the test is classified as inapplicable.
- Each Class E test is self-checking and produces a NOT APPLICABLE, PASSED, or FAILED message when it is compiled and executed. However, the LRM permits an implementation to reject programs containing some features addressed by Class E tests during compilation.

Therefore, a Class E test is passed by a compiler if it is compiled successfully and executes to produce a PASSED message, or if it is rejected by the compiler for an allowable reason.

- Class L tests check that incomplete or illegal Ada programs involving multiple, separately compiled units are detected and not allowed to execute. Class L tests are compiled separately and execution is attempted. A Class L test passes if it is rejected at link time – that is, an attempt to execute the main program must generate an error message before any declarations in the main program or any units referenced by the main program are elaborated.

The test-naming conventions used in the ACVC is based on the ACVC *Implementors' Guide* (AIG):

- The initial character is the test class.
- Characters 2 through 4 comprise the AIG chapter, section, and subsection numbers.
- Characters 5 and 6 compose the AIG test objective number.
- Character 7 is a test sequence number.
- Character 8 (if present) is a test file sequence number (where a test comprises more than one file).

In multiple-file tests, a ninth character, M, denotes the main program. The AIG's chapter/section/subsection numbers follow those of the LRM.

The results of the ACVC test suite provide the raw information concerning conformity of the compiler to the LRM and its capabilities as demonstrated by results from certain tests and as documented in the VSR.

2.1 THE VALIDATION SUMMARY REPORT

The VSR is prepared by the Ada Validation Facility (AVF) and published upon the successful

completion of the validation effort. Table 1 is a depiction of a VSR outline.

TABLE 1. VSR Outline

1. INTRODUCTION
   - 1.1 Purpose of This Validation Summary Report
   - 1.2 Use of This Validation Summary Report
   - 1.3 References
   - 1.4 Definition of Terms
   - 1.5 ACVC Test Classes
2. CONFIGURATION INFORMATION
   - 2.1 Configuration Tested
   - 2.2 Implementation Characteristics
3. TEST INFORMATION
   - 3.1 Test Results
   - 3.2 Summary of Test Results by Class
   - 3.3 Summary of Test Results by Chapter
   - 3.4 Withdrawn Tests
   - 3.5 Inapplicable Tests
   - 3.6 Test Modifications
   - 3.7 Additional Testing Information
     - 3.7.1 Prevalidation
     - 3.7.2 Test method
     - 3.7.3 Test suite

The VSR documents the results of formal validation testing. In addition, the VSR records allowable implementation system dependencies observed during the testing process. These dependencies include characteristics that are defined by the compiler, such as the maximum length of variable names or the maximum precision of floating point numbers. Thus, the VSR provides a primary resource for obtaining information on compiler capabilities.

Two sections of the VSR describe compiler characteristics: Section 2.2, "Implementation Characteristics," and Section 3.5, "Inapplicable Tests." Generally, a test is inapplicable if it tests some implementation-specific feature of the language that is not supported by the compiler under test. Thus, a compiler with a large number of inapplicable tests may be considered less capable than one with few inapplicable tests.

2.2 COMPILER DESCRIPTION - APPENDIX F

The LRM requires that a compiler vendor supply an explicit description of all implementation characteristics as part of the compiler documentation. The vendor supplies an Appendix F as specified in the LRM to the AVF as part of the documentation required for the validation process. Some dependencies are tested in the ACVC and recorded in the VSR. However, there are dependencies, especially with respect to pragmas and attributes, whose description can only be found in the vendor's documentation. Thus, a vendor's Appendix F, which is included in the VSR, also provides useful information.

The LRM requires that an implementation's reference manual include the following information in its Appendix F:

- The form, allowed places, and effect of every implementation-dependent pragma.
- The name and the type of every implementation-dependent attribute.
- The specification of the package SYSTEM.
- The list of all restrictions on representation clauses.
- The conventions used for any implementation-generated name denoting implementation-dependent components.
- The interpretation of expressions that appear in address clauses, including those for interrupts.
- Any restriction on unchecked conversions.
- Any implementation-dependent characteristics of the input-output packages.

In practice, most vendors' Appendix Fs use a structure guided by this list where section headings correspond to the list. The amount of detailed information provided varies from vendor to vendor.

3. SPECIFIC SOURCE OF COMPILER CAPABILITY INFORMATION

The following sections discuss sources of capability information that is generated from the validation process.

3.1 CAPACITY OF THE CONFIGURATION

Compiler capacity is an attribute that is limited by available resources, e.g., the maximum recursion. Thus, capacity is usually dependent on the resources available in the target computer.

3.1.1 Maximum Number of Variables

One measure of a compiler's capacity is the number of distinct variable names that can occur in a single compilation unit. A test for this type of capacity is a side effect of test C29002K.

3.1.2 Maximum Nesting Levels

The ACVC tests three types of nesting levels: loops, recursion, and blocks. The tests involve families of tests that test up to several specific levels. The successful execution of the maximum nesting depth may be limited by available memory in the target computer rather than by the compiler.

3.2 COMPILER CAPABILITIES

The capabilities described here are compiler attributes which are present or absent, such as PRAGMA INTERFACE, or can be quantified with a simple integer, such as the number of digits of precision in floating point numbers.

3.2.1 Arithmetic

The VSR reports various facts about the arithmetic characteristics of the subject implementation. These facts include:

- The decimal precision of each predefined floating-point type;
- The range of each predefined integer type;

- The observed method(s) of rounding (floating-point values) to integer;
- A general indication of support for fixed-point types; and
- A general indication of out-of-range integer calculations and how underflow is handled.

This information is reported in the VSR Sections 2.2, 3.5, and Appendix B.

Appendix B of the VSR contains both the Appendix F provided by the Vendor as well as the declarations of the predefined numeric types (from package STANDARD). The vendor's Appendix F contains the specification of package SYSTEM, and so gives the implementation-dependent values MAX_INT, MIN_INT, MAX_DIGITS, MAX_MANTISSA, and FINE_DELTA.

Whether an implementation has the capability to process integer calculations with transient values that exceed MAX_INT is checked by tests D4A002A&B (for 32-bit values) and D4A004A&B (for 64-bit values). The results are reported in the VSR Section 2.2.

Two series of tests check whether an implementation can support either "fine" or "coarse" fixed-point types requiring various bit lengths. The limitations of an implementation are indicated by the particular tests that are listed in the VSR Section 3.5 as Not Applicable (NA).

A number of ACVC tests check how a compiler carries out expression evaluation. The results of these tests are recorded in the VSR Section 2.2, "Implementation Characteristics."

Ada allows the programmer to declare a floating point subtype with less precision than the base type. Assignments to variables of the less precise subtype may be performed at the precision of the (more precise) base type or the (less precise) subtype. ACVC test C35712B checks to see what precision an implementation uses.

Some computer systems provide extra bits for numeric calculations. This is more a capability of the computer hardware than the compiler. Nevertheless, the capability would be of interest to programmers of systems requiring highly precise calculations. ACVC program C35903A checks for

this behavior for fixed-point calculations.

Another capability that is more related to the underlying hardware is how underflow is treated. A system may allow gradual underflow by using guard bits or by denormalizing the exponent during floating-point calculations. This behavior is checked with ACVC tests C45524A through C45524Z.

Computer systems use a variety of methods for rounding integers. The methods include rounding toward positive infinity, negative infinity, or zero. The integer rounding technique is checked in ACVC tests C46012A through C46012Z. Test C4A014A determines the mechanism for rounding an integer in a static real expression.

3.2.2 Representation Clauses

Representation clauses allow programmers to control how objects are stored in memory. Enumeration representation clauses explicitly specify how enumeration types are represented. This capability may be required, for example, when an enumeration variable must be passed to a subprogram written in a language other than Ada. There are fourteen ACVC programs that make use of enumeration representation clauses. These are: C35502I, C35502J, C35502M, C35502N, C35507I, C35507J, C35507M, C35507N, C35508I, C35508J, C35508M, C35508N, C55B16A, and A39005F.

Length clauses are used to specify how much memory is to be allocated for an object. There are three ways in which this may be specified. The following describes these constructs and the related ACVC tests:

- SIZE specifies an upper bound on the number of bits to be allocated to objects of the specified type. The relevant ACVC programs are C87B62A and A39005B.
- The STORAGE_SIZE specification is used with access types to indicate the total amount of storage to be allocated for all objects that are to be generated. STORAGE_SIZE may also be

given for task types to specify the number of storage units to reserve for a task. The ACVC tests for access types are C87B62D and A39005D.

- The SMALL specification is used to indicate how fixed point numbers are represented. The ACVC tests that test the SMALL representation clause are C87B62C and A39005E.
- Record representation clauses are used to specify how record types are stored including the order, position, and size of record components. It is not possible for an Ada program to check whether record representation clauses have been correctly processed – a manual examination of storage contents would be necessary. Thus, the ACVC only contains tests that check that these clauses may be given only in the allowed contexts. These tests will be introduced in ACVC version 1.10.

3.2.3 INLINE Pragma

The INLINE pragma is used to indicate that the code generated by a subprogram should be expanded as a macro. Implementors may choose whether or not to provide this capability. As with other language defined capabilities, the ACVC includes programs that make use of the INLINE pragma. Thus, the absence of this capability in a specific compiler will be noted by the reference to inapplicable tests in the VSR. In addition, the VSR includes an explicit statement as to whether the INLINE pragma is supported. The relevant tests are: LA3004A, LA3004B, EA3004C, EA3004D, CA3004E, and CA3004F.

3.2.4 Files

The primary input/output mechanism for Ada is the file. Chapter 14 of the LRM defines three primary file types: direct, sequential, and text. The definitions include the procedures and functions to be invoked to open, close, read, write, and interrogate the various file types. However, compilers differ in their file management. Some compilers have no file capabilities at all: these are compilers whose target machines are to be embedded within some special purpose system, such as a missile,

where there is no need for file management. Others have complete file management support including the ability of having multiple internal files access a single external file.

Another way in which compilers differ is in the supported file modes. Files may be opened for read access (mode IN_FILE), write access (mode OUT_FILE), or both (mode INOUT_FILE). The ACVC includes a number of programs that ensure that an exception is raised in cases where a specific file mode is not supported.

The ACVC also includes programs that test for exceptions when specific operations are not supported. ACVC test CE2102G checks for RESET and DELETE operations on SEQUENTIAL_IO files and CE2102K checks for RESET and DELETE operations on DIRECT_IO files.

Some applications, such as transaction processing systems, may require that several internal files be associated with a single external file. A number of ACVC programs test how a compiler handles multiple internal representations of a single external file. Tests CE2107A, CE2107C, CE2107F, and CE2107I determine whether multiple internal files may be generated for a single sequential or direct access file. Tests CE2107B, CE2107D, CE2107G, and CE2107H are applicable if different internal files may separate different element types for the same file. Test CE2107E is applicable if the same external file may be represented by a direct and sequential internal file.

A series of ACVC programs determines how a compiler handles combinations of read and write accesses for multiple internal files associated with a single external text file. Program CE3111A tests for multiple open operations for read access; program CE3111B tests for multiple open operations, one of which is for write access; and program CE3111C tests for multiple open operations, two of which are for write access.

It should be noted that the LRM does not indicate how files with multiple internal representations should be protected when a write operation may conflict with other read or write operations.

Specifically, the LRM does not mention any type of locking mechanisms for files or records. A description of protection mechanisms for files with concurrent accesses should be found in Appendix F of an implementation's reference manual.

Some compilers might provide additional flexibility in file management by allowing the specification of files with unconstrained data types. ACVC programs AE2101C and AE2101H test for the instantiation of sequential and direct access files with unconstrained arrays, and tests EE2101D and EE2401D test for READ, WRITE, and END_OF_FILE operations on those files. Tests EE2201E and EE2401G test for READ, WRITE, and END_OF_FILE operations on sequential and direct access files with variant record types with non-default discriminants.

3.2.5 Interrupts

The provision of a capability to receive and process interrupts is implementation dependent. Generally, Ada compilers that compile programs for execution on large, time-shared computer systems do not need, and usually do not want, such machine-level capabilities available to applications programmers. However, programs that are intended to operate in a real-time embedded environment will usually require the correct processing of interrupts.

The LRM provides a general description of how an interrupt is to be handled in Section 13.5.1. However, the ACVC test suite does not currently contain tests for interrupts. The rationale for the absence of interrupt tests is based on the implementation-dependent nature of interrupts.

3.2.6 Generic Compilations

Most of the previous discussion centered on the capabilities that a compiler provides to the executable system. Another capability that differs among compilers is the separate compilation of a generic unit's declaration and body. Some compilers require that a generic unit be submitted in a single compilation (file); some compilers allow separate compilation so long as no instantiations of the generic unit occur prior to the body; still others prohibit any instantiations of the generic unit

prior to the body – even within the same compilation.

Although this has no effect on the capabilities of the generated code, it may have a significant influence on the program development environment. A change to a generic body requires recompilation of the body and all other programs dependent on the body. If the declaration and body are located in the same file, then the program library will detect the recompilation of the generic declaration. Thus, all programs dependent on the generic specification unit will require recompilation as well.

The following ACVC programs are listed as inapplicable if a generic declaration and body must reside in a single file: CA1012A, CA2009C, CA2009F, BC3204C, BC3205D, and CA3011A.

3.2.7 Split Tests

Section 3.6 of the VSR discusses test modifications required during validation. Generally, test modifications are limited to the splitting of tests. Historically, two reasons have been accepted for splitting a test. First, a B-test may contain multiple errors and the compiler stops upon encountering the first error. Second, a compilation unit may be too large for the configuration.

As in the discussion of generic compilation units above, the ability to detect multiple errors is an attribute of a compiler which does not effect the executable system. However, this capability can significantly decrease the amount of time required to eliminate syntactic errors in a program under development.

The need to split tests because of the compilation unit size may be due to limitations of the compiler system, the host computer resources, or the target computer resources. Thus, the justifications for splitting tests must be read carefully to determine whether or not this action was necessary because of limitations of the compiler.

4. CONCLUSIONS AND RECOMMENDATIONS

Validation testing exposes a variety of implementation-dependent information that is of interest to potential buyers and users of a compiler. VSRs report this information in a standard, easy-to-read format. The detailed results of validation testing are reported in VSR Sections 2.2, "Implementation Characteristics," and Section 3.5, "Inapplicable Tests." Further information about implementation characteristics is presented by the vendor in the VSR Appendix B, which contains the package STANDARD's numeric type declarations and the implementation's Appendix F.

The Appendix Fs supplied by vendors differ in both detail, structure, and presentation. However, this is to be expected as the LRM makes only a general requirement of vendors with respect to Appendix F. So long as the required information is prsent in some form, a vendor has satisfied the requirements of the LRM and of validation.

It is recommended that no further requirements be imposed on vendors to produce documents of a particular format to specify compiler capabilities. Such a requirement would increase the cost of validation for both vendors, the AVF staff, and the Ada Validation Organization personnel, and has a dubious legal basis, given the commercial importance of validation to vendors. The exact requirements of the LRM with respect to implementation-dependent language features defined in Chapter 13 of the LRM are currently the focus of the Ada Rapporeteur Group under ISO WG9 in its deliberations of Commentary AI-00361. Depending on the outcome of these deliberations and approval by the AJPO, new ACVC tests may provide further implementation-dependent information that will be presented in the VSR Sections 2.2 and 3.5. Improvements in the presentation of implementation characteristics is an on-going effort of the AVFs, Ada Validation Organization, and ACVC Maintenance Organization.

Distribution List for IDA Memorandum Report M 454

NAME AND ADDRESS	NUMBER OF COPIES
Sponsor	
Ms. Virginia Castor Ada Joint Program Office 1211 Fern St., Room C-107 Arlington, VA 22202	2 copies
Other	
Defense Technical Information Center Cameron Station Alexandria, VA 22314	2 copies
IIT Research Institute 4550 Forbes Blvd., Suite 300 Lanham, MD 20706 (AJPO documents only)	1 copy
IDA	
Ms. Audrey A. Hook, CSED Mr. R. Danford Lehman, CSED	2 copies 2 copies

END
DATE
9-88
DTIC